VBA: Rubbersheet

Contributed by Bert Granberg
16, May. 2008
Last Updated 16, May. 2008

This script was written by Tim Hodson ofESRI to make wholesale rubbersheet-style adjustments to all the layers in an edit session based on a shift in location of control points. It was written to adjust parcels, sections, and other boundary features based on old GCDB control points to line up with new, more accurate and/or precise GCDB control points obtained with survey grade GPS. It requires that the old and new points are stored in separate layers but carry common unique IDs for each point. Thanks Tim!

```
Option Explicit
Private Sub RubbersheetEditSessionLayersinMapBasedOnPointShift()

    Dim pEd As IEditor, pLyr As ILayer, pFeatLyr As IFeatureLayer
    Dim pTr As ITransformationMethod, pEnv As IEnvelope2, pFWS As IFeatureWorkspace
    Dim pGeoDataset As IGeoDataset, pFeatCurs As IFeatureCursor
    Dim pOldPointsFC As IFeatureClass, pNewPointsFC As IFeatureClass
    Dim pAdjFeatClass() As IFeatureClass, LayerName As String
    Dim MatchFieldName As String, OldControl As String, NewControl As String
    Dim SearchWithinTolerance As Double, lCnt As Long, lCnt2 As Long
    Dim pEnumLyr As IEnumLayer, pDS As IDataset, pActView As IActiveView
    Dim pMXDoc As IMxDocument

    'Here's some code that will update the feature layers based on a shift from
    'the old GCDB control to the new control.
    '
    'It has the following requirements:
    '----------------------------------
    'The edit workspace must have two point feature classes that represent the old
    'and the new control points.
    'These point feature classes must both have attribute fields (of type string)
    'that can be used to identify common points in each feature class.
    'The name of the field must be the same in both feature classes
    'The data and map frame should be in a projected coordinate system
    'Point IDs only need to be unique within the tolerance distance specified,
    '(I used 50 meters)
    'The layers to be adjusted must be in the map and in the same edit workspace.
    '(remove layers in the workspace that should not be moved)
    'The point feature classes do not need to be added to the map, but if they
    'are they are not adjusted (the old control points do not move)
    'You must start editing on the workspace that has the layers to be adjusted.
    '
    'If needed, change the parameters using the 4 lines of code, as described below:

    'Parameters
    '----------
    MatchFieldName = "pntid" ' this is the common name of the field used in both
                             ' the point feature classes
    SearchWithinTolerance = 150 ' point matches will only occur within this tolerance
                                ' distance from the old control point (units are the
                                ' same as those of the projected coordinate system)
    OldControl = "GCDBCoor_Old" ' The name of the feature class for the old control pts
    NewControl = "GCDBCoor_New" '' The name of the feature class for the new control pts

    On Error GoTo handler

    Set pEd = Application.FindExtensionByName("esri object editor")

    If pEd.EditState = esriStateNotEditing Then
    MsgBox "Please Start Editing"
    Exit Sub

End If

lCnt = 0

Set pFWS = pEd.EditWorkspace
Set pOldPointsFC = pFWS.OpenFeatureClass(OldControl)
Set pNewPointsFC = pFWS.OpenFeatureClass(NewControl)

If pOldPointsFC.ShapeType <> esriGeometryPoint Or _
    pNewPointsFC.ShapeType <> esriGeometryPoint Then

      MsgBox "Two point feature classes are requireed."
      Exit Sub

End If

Set pGeoDataset = pOldPointsFC
Set pEnv = pGeoDataset.Extent

Set pEnumLyr = pEd.Map.Layers
pEnumLyr.Reset
Set pLyr = pEnumLyr.Next

While Not pLyr Is Nothing

    LayerName = pLyr.Name

    If TypeOf pLyr Is IFeatureLayer Then

        Set pFeatLyr = pLyr
        Set pDS = pFeatLyr.FeatureClass

        If LayerName <> OldControl And LayerName <> NewControl And _
            ObjPtr(pDS.Workspace) = ObjPtr(pEd.EditWorkspace) Then

            ReDim Preserve pAdjFeatClass(0 To lCnt)
            Set pAdjFeatClass(lCnt) = pFeatLyr.FeatureClass
            Set pGeoDataset = pAdjFeatClass(lCnt)
            pEnv.Union pGeoDataset.Extent
            lCnt = lCnt + 1
        End If

    End If

    Set pLyr = pEnumLyr.Next

Wend

Set pTr = GetRubberSheetTransformation(pOldPointsFC, pNewPointsFC, MatchFieldName, _
          SearchWithinTolerance, pEnv)

pEd.StartOperation

For lCnt = 0 To UBound(pAdjFeatClass)

    Set pFeatCurs = pAdjFeatClass(lCnt).Update(Nothing, False)
    Debug.Print "Transforming features in " & pAdjFeatClass(lCnt).AliasName & "..."
    Debug.Print Time$
    pTr.Transform pFeatCurs, Nothing
    Debug.Print "Completed transforming features in " & _
            pAdjFeatClass(lCnt).AliasName & "..."
    Debug.Print Time$
    Next lCnt

    pEd.StopOperation "Feature Update by Rubbersheet"

    Set pMXDoc = Document
    Set pActView = pMXDoc.FocusMap
    pActView.PartialRefresh esriViewGeography, Nothing, pActView.Extent
    Debug.Print "Completed: " & Time$

    Exit Sub

handler:

    MsgBox Err.Description
    pEd.AbortOperation

End Sub

Private Function GetRubberSheetTransformation(SourcePointFC As IFeatureClass, _
              TargetPointFC As IFeatureClass, MatchFieldName As String, _
              Tolerance As Double, Envelope As IEnvelope2) As ITransformationMethod

    Dim pFromPts() As IPoint, pToPts() As IPoint, lSegCnt As Long
    Dim pOldPointFeature As IFeature, pNewPointFeature As IFeature
    Dim pFromPoint As IPoint, pToPoint As IPoint, pTr As ITransformationMethod
    Dim sNameID As String, pFeatCursOuter As IFeatureCursor, pFeatCursInner As IFeatureCursor
    Dim pQueryFilter As IQueryFilter, pLine As ILine2
    Dim pWorkspace As IWorkspace, pSQLSyntax As ISQLSyntax
    Dim sDelimPrefx As String, sDelimSuffx As String

    Set pFeatCursOuter = SourcePointFC.Search(Nothing, False)
    Set pOldPointFeature = pFeatCursOuter.NextFeature
    Set pLine = New Line

    While Not pOldPointFeature Is Nothing

        sNameID = pOldPointFeature.Value(pOldPointFeature.Fields.FindField(MatchFieldName))
        Set pQueryFilter = New QueryFilter
        pQueryFilter.WhereClause = MatchFieldName & "='" & sNameID & "'"
        Set pFeatCursInner = TargetPointFC.Search(pQueryFilter, False)
        Set pNewPointFeature = pFeatCursInner.NextFeature

        While Not pNewPointFeature Is Nothing

            Set pFromPoint = pOldPointFeature.ShapeCopy
            Set pToPoint = pNewPointFeature.ShapeCopy
            pLine.PutCoords pFromPoint, pToPoint

            If pLine.Length <= Tolerance Then

                ReDim Preserve pFromPts(0 To lSegCnt)
                Set pFromPts(lSegCnt) = pFromPoint
                ReDim Preserve pToPts(0 To lSegCnt)
                Set pToPts(lSegCnt) = pToPoint
                lSegCnt = lSegCnt + 1

            End If

            Set pNewPointFeature = pFeatCursInner.NextFeature

        Wend

        Set pOldPointFeature = pFeatCursOuter.NextFeature
    Wend

    Set pTr = New PiecewiseTransformationMethod

    If lSegCnt > 0 Then

        pTr.DefineFromControlPoints lSegCnt, pFromPts(0), pToPts(0), Nothing, Envelope
        Set GetRubberSheetTransformation = pTr

    End If

End Function
```